28 | 网络数据传输慢，问题到底出在哪了？

2020-01-29 庄振运

性能工程高手课 进入课程 >

讲述：庄振运
时长 15:38　大小 12.54M

你好，我是庄振运。

你一定有过在网页或者手机上下载照片的体验，如果数据传输太慢，那你的体验一定十分糟糕。你看，互联网实体之间的数据快速传输对用户体验至关重要。这里涉及到的其实就是网络传输问题。所以，今天我们就通过生产实践中的案例，来探讨一下互联网服务中的数据传输性能。

说到底，网络传输问题其实就分两种：

1. 数据根本没有传递；

2. 数据传送速度较慢。

“数据没有传递”虽然看起来更严重，但是相对“数据传送缓慢”来说，更容易判断和解决。所以，这一讲，我们就重点解决第二种问题。我们一起来看看，为什么网络传送速度会慢，在众多原因中怎么快速诊断出关键问题来，又该如何去解决。

造成网络传输缓慢的原因很多，我们这一讲，就是帮助你快速诊断问题出在哪里：是客户端，是服务器端，还是网络本身？在此基础上，你才能专门针对具体的领域继续分析。

为什么数据传输慢？

我们先看一下，都有哪些可能的原因会导致数据传输缓慢呢？在宏观上，这种问题的可能原因可以分为三种场景：

1. 客户端应用程序的原因；
2. 网络的原因；
3. 服务器应用程序的原因。

也就是说，可能是由于数据发送方过载，而没有向接收方发送数据；也可能是网络通道很慢；又或者是数据接收方的服务器太忙，从而无法从网络缓冲区读取数据。

为了描述方便，我们根据平时客户浏览网页的场景，假设客户端是数据接收方，而服务器端是数据发送方。

进行此类分析诊断时，负责的工程师通常需要快速隔离出上述不同场景，以便他们可以专注于特定场景里面的可疑组件，并对本质原因进行更深入的分析。但这一快速诊断过程会遇到很多难点。

首先，**数据传输涉及多个网络实体**，包括两台机器（也就是发送者和接收者）和网络路由，这与仅涉及一台机器的常见性能问题形成鲜明对比。

其次，这种**诊断涉及多层信息**，包括应用程序层和网络传输层。为了找出原因，工程师必须检查各种数据，包括客户端日志、服务器日志、网络统计信息、CPU 使用情况等。这些检查需要花费很多时间和精力，并且通常需要性能工程师的经验和专业知识。

更加让人郁闷的是，这些日志往往分散在不同的地方，比如客户端和服务器。为了节省时间和精力，性能工程师迫切需要更智能的工具，以帮助他们快速找出根本原因。

所以今天，我专注于解决这样的一个问题，就是：快速确定应归咎的组件范围和场景（无论是发送方、接收方还是网络本身）。我提出了一种**当发生数据传输缓慢的问题时，可以自动隔离原因的解决方案**。毕竟，你只有找出了要对“数据传输慢”负责任的那一部分，才可以进行后续分析工作，最终确定真正的问题。

如何判断问题所在位置？

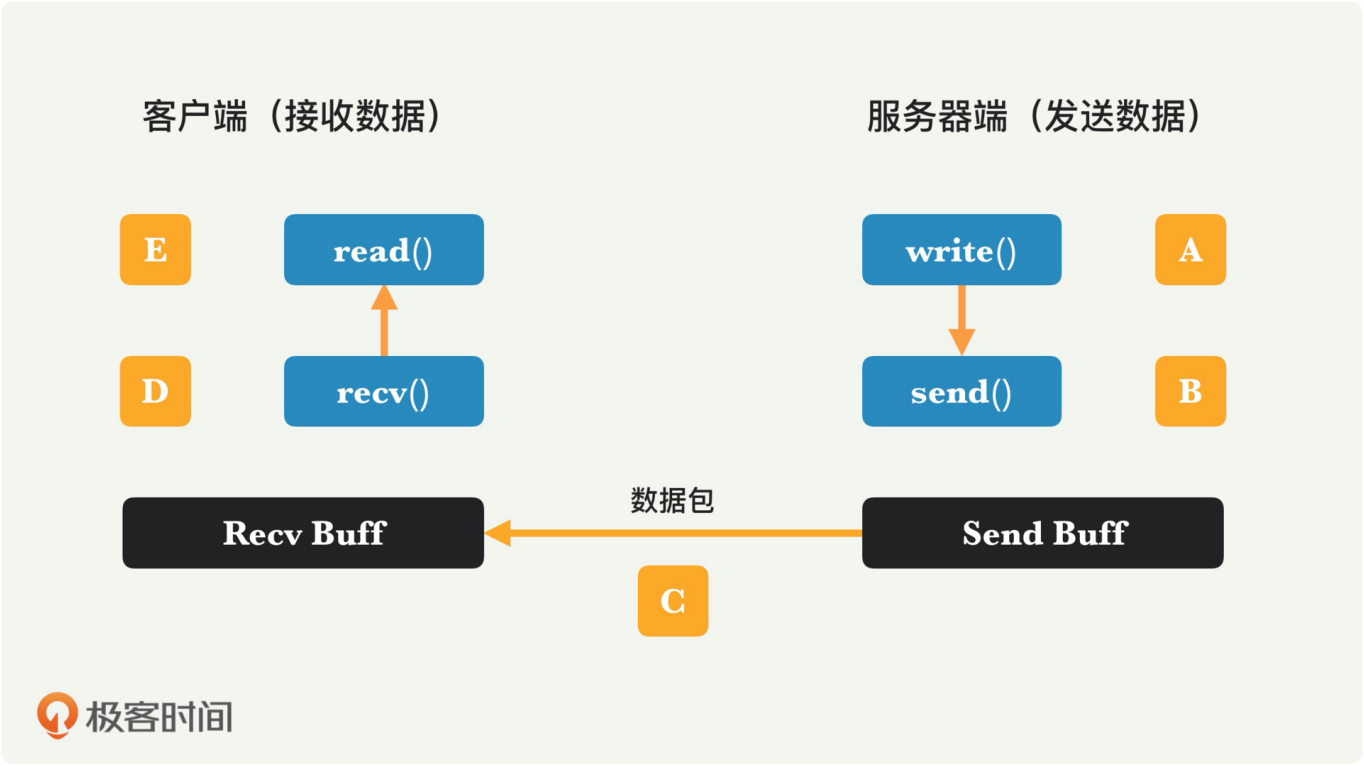
要想快速诊断，我们需要先看看三种问题场景的不同特征。

这个解决方案本质上依靠的是**客户端和服务器端的 TCP 层面的特征**。TCP 是传输层协议之一，可提供有序且可靠的流字节传输，是当今使用最广泛的传输协议。TCP 具有**流控制**功能，可避免接收方过载。接收方设置专用的接收缓冲区，发送方设置相应的发送缓冲区。数据发送方（服务器）的发送缓冲区和数据接收方（客户端）的接收缓冲区，都可以通过操作系统来监测当前队列大小。

为了能够识别瓶颈，你需要在发送方和接收方的传输层上，收集有关队列大小的信息。有很多收集此类信息的方法。你有两种工具可以使用，分别是 Netstat 和 ss。

Netstat 是一个命令行工具，可以显示网络连接和网络协议统计信息。我们主要是用它来观察 TCP / IP 套接字的发送队列和接收队列的大小。而 ss 命令，可以显示套接字统计信息，包括显示 TCP 以及其他类型套接字的统计信息。类似于 Netstat，ss 还可以显示发送和接收队列大小。

除了相应的工具的介绍，为了帮助你理解，我们还需要先重温一下传输数据时候，应用层和 TCP 层的交互。

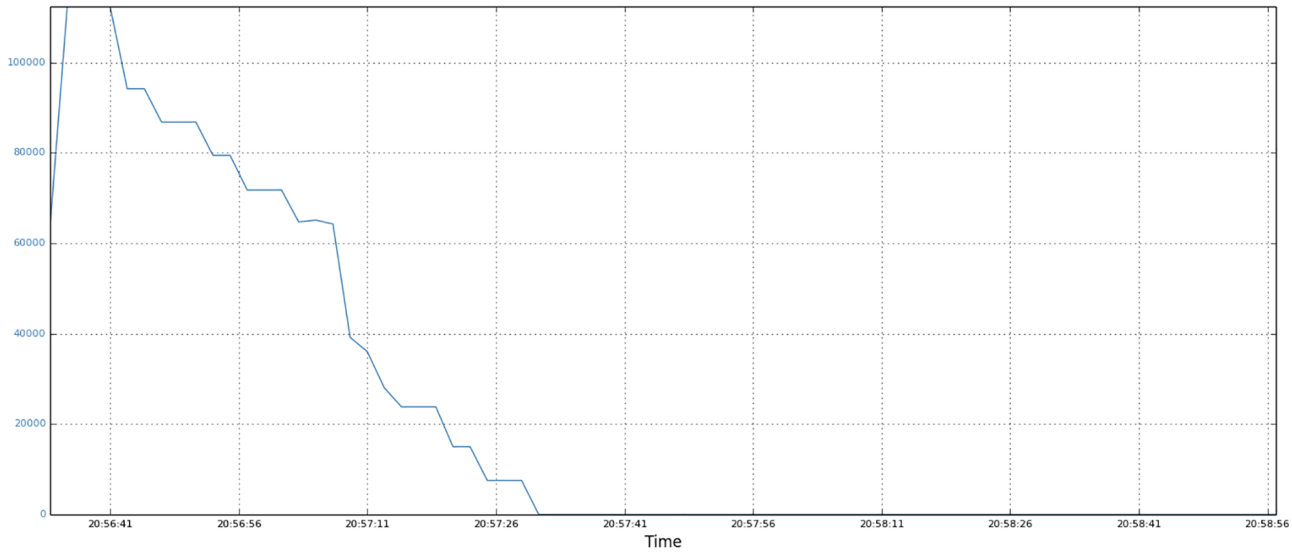


上图显示了任何基于 TCP 的数据传输中的典型流程。关于系统调用和网络传输的五个步骤如下：

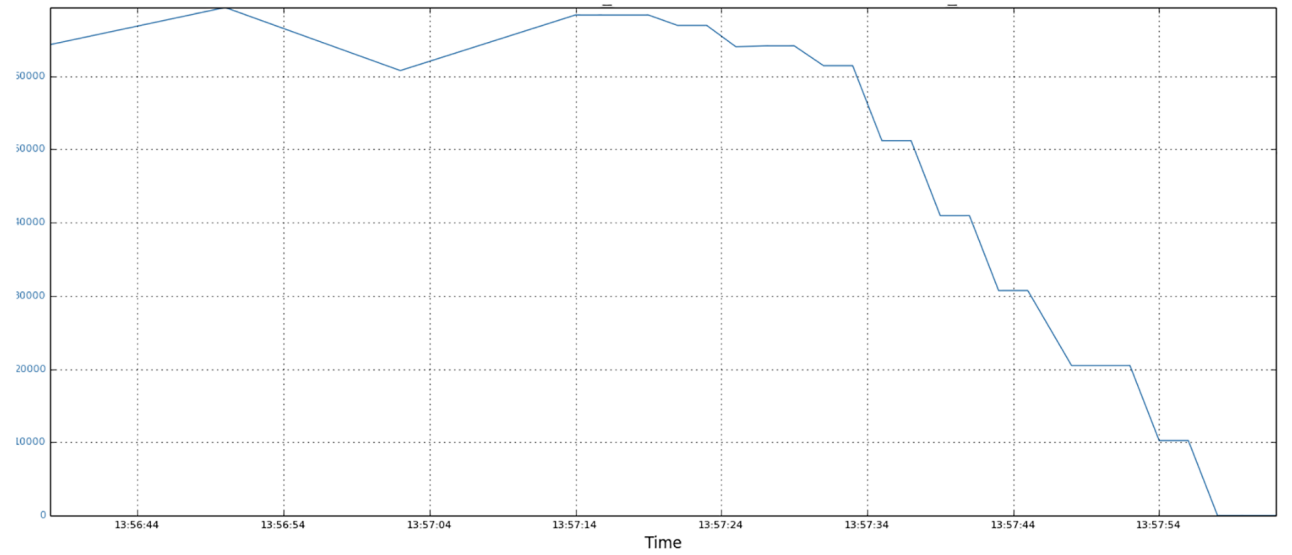
1. 在步骤 A，服务器应用程序发出 write() 系统调用，并将应用程序数据复制到套接字发送缓冲区。
2. 在步骤 B，服务器的 TCP 层发出 send() 调用，并将一些数据发送到网络；数据量受 TCP 的拥塞控制和流控制。
3. 在步骤 C，网络将数据逐跳路由到接收方（IP 路由协议在这部分中发挥作用）。
4. 在步骤 D，客户端的 TCP 层将通过 recv() 系统调用接收数据，数据放入接收缓冲区。
5. 在步骤 E，客户端应用程序发出 read() 调用，以接收数据并将其复制到用户空间。

接下来，我们就来看看三种不同场景下的问题特征是什么样的。

我们先看第一个场景，**客户端接收数据缓慢**的情况。为了重现这一场景，我们做一个实验，让发送端发送一段固定大小的数据给接收方。我们强制接收方，也就是客户端，减慢数据的接收速度。具体做法，就是在应用程序代码的 read() 调用之前，注入了一定的延迟，这种场景代表了客户端数据接收成为瓶颈的情况。

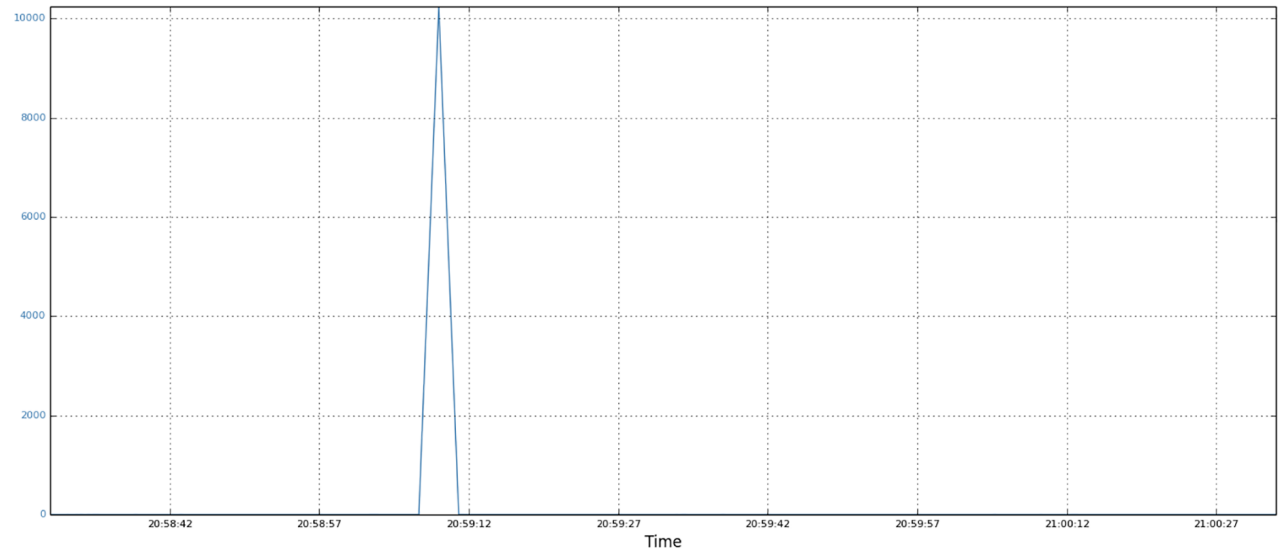


上图显示的是数据发送方的发送缓冲区，SendQ（Send Queue）的大小变化。开始时候，数据发送调用 send()，立刻注满 SendQ。随着数据的传输，慢慢变为 0。



第二张图是客户端的接收缓冲区，RecvQ（Receive Queue）的大小变化，客户端因为应用程序运行缓慢，所以 RecvQ 具有一定的积累，这可以由非零值来看出。这些非零值持续了一段时间，随着应用程序不断地读取，最终 RecvQ 减为 0。

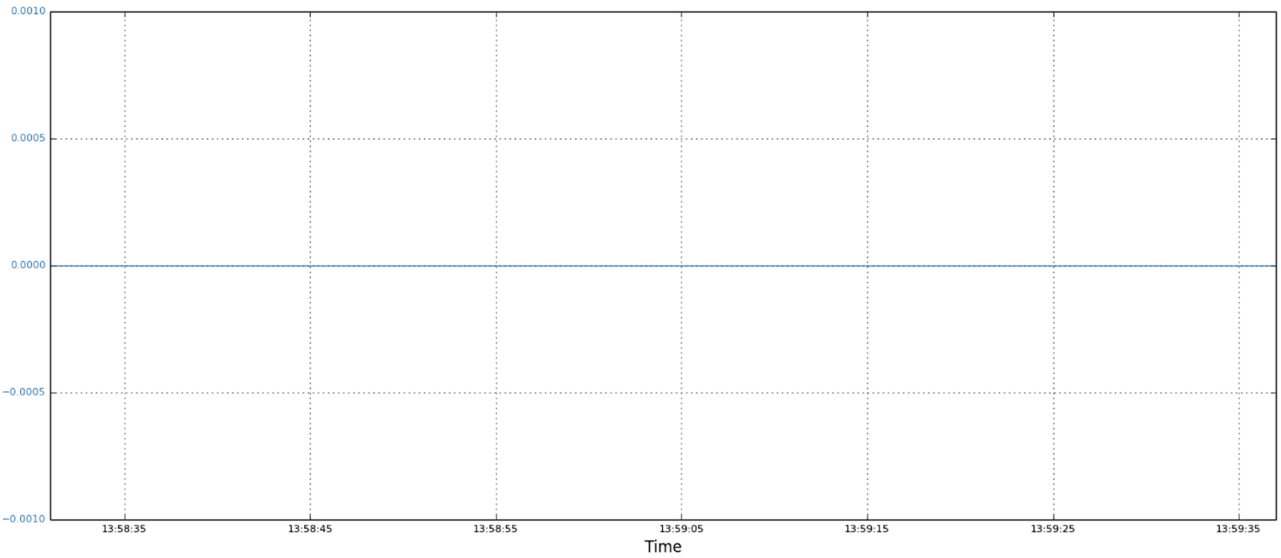
对于第二种场景，也就是**数据发送方是瓶颈**的情况，我们强制发送方（即服务器端），放慢数据的发送速度。具体来说，我们在应用程序代码中，对 write() 的调用之前注入了一定的延迟，模拟了发送者是瓶颈的情况。



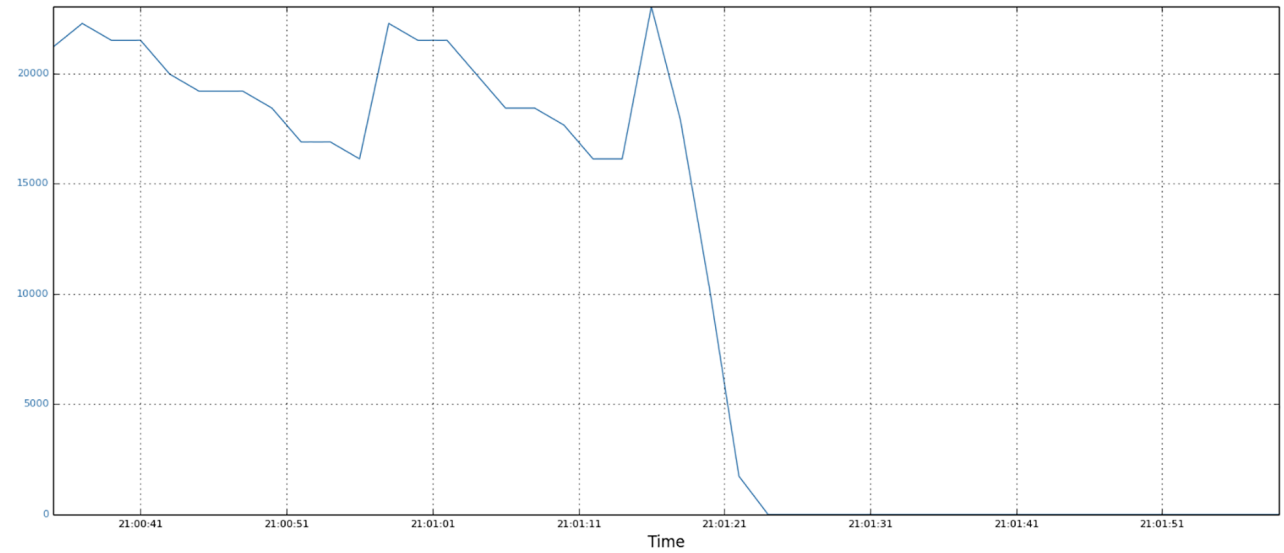
上图显示了服务器端的 SendQ 的值，你可以看到，SendQ 几乎全部是零。这是因为发送端是瓶颈，其他地方不是瓶颈，所以任何 SendQ 的数据会被很快发送出去。

你可以在图片中看到一个持续时间很短的峰值，这是因为 SendQ 取样的时候恰好取到数据还没有被传输到网络中的时候。但因为这个峰值持续时间很短，简单的过滤就可以去掉。

接收端的 RecvQ 显示在下图，你可以看到，因为接收端不是瓶颈，RecvQ 是零。

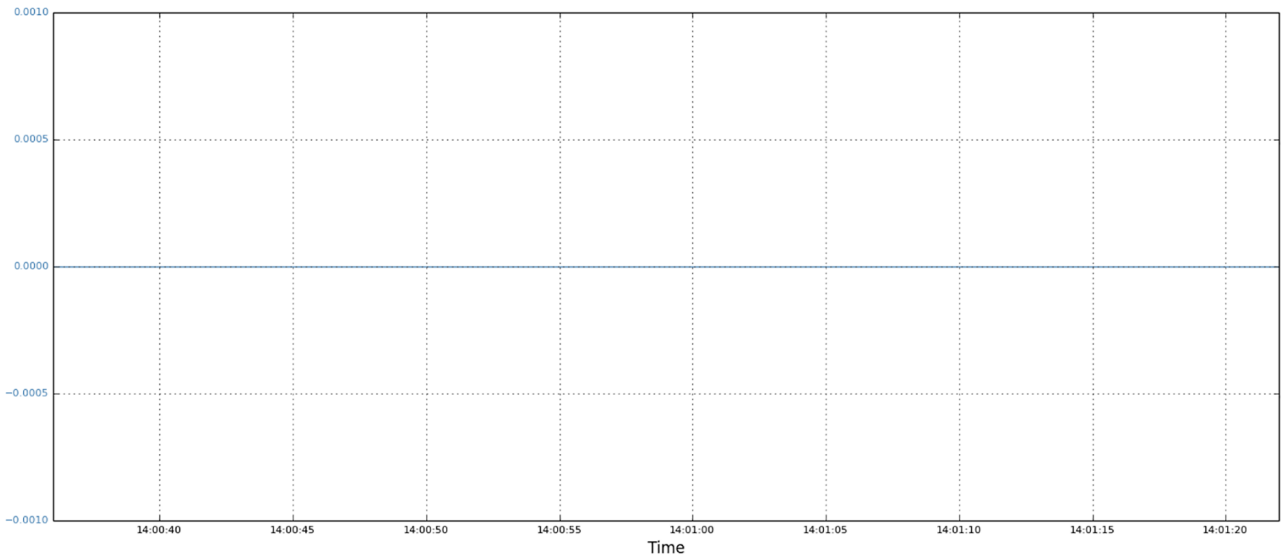


第三个场景，是**网络本身是瓶颈**造成的数据传输缓慢。我们通过向网络路径注入延迟来创造这一场景，以使 TCP 仅能以非常低的吞吐量进行传输。



图片中显示了发送端的 SendQ 值，你可以看到它的值不为零，因为那些数据不能很快地被传送出去。

再来看接收端的 RecvQ，如下图。RecvQ 全为零，这些零值就代表了快速的数据传递。



通过上面三种场景的分析，尤其是对发送端 SendQ 和接收端 RecvQ 的观察，我们不难总结出规律来。正常的数据传输情况下，客户端的接收队列和服务器端的发送队列都应该是零。

反之，如果数据传输缓慢，则有如下几种情况：

1. 如果客户端上的接收队列 RecvQ 不为零，则客户端应用程序是性能瓶颈；
2. 如果服务器上的发送队列 SendQ 为零，则服务器应用程序是性能瓶颈；
3. 如果客户端的接收队列 RecvQ 为零，而服务器的发送队列 SendQ 为非零，则网络本身是性能瓶颈。

为了帮助你加深记忆，我用表格来做了个归纳。

性能瓶颈所在	接收队列 RecvQ	发送队列 SendQ	说明
客户端（接收方）	非0	任意	数据消费太慢
服务器端（发送方）	任意	0	数据准备太慢
网络本身	0	非0	网络传输太慢

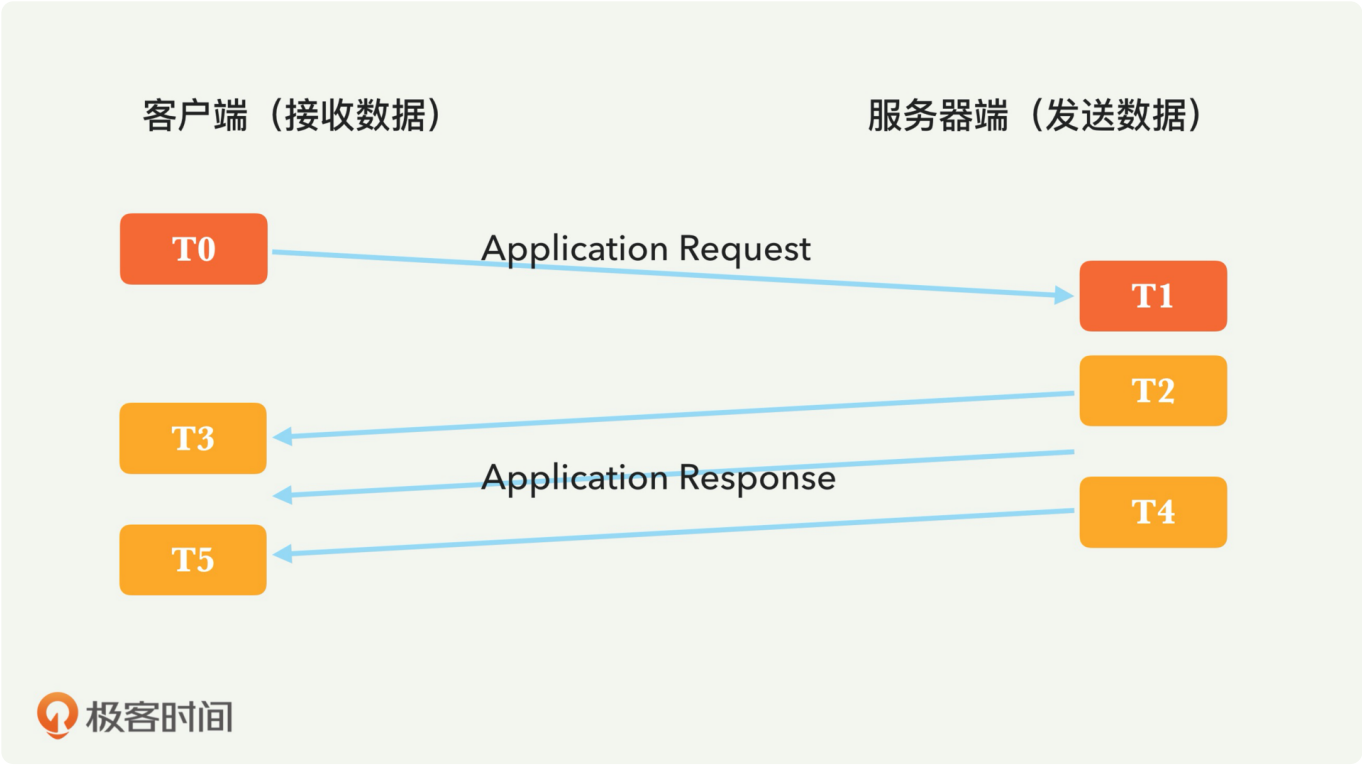
极客时间

你可以通过这个表格，快速判断问题出现的位置。

解决方案如何落地？

根据前面的分析和总结，我们现在提出解决方案。这是一个基于**状态转移**的方案，需要从客户端和服务器端收集几个关键点的信息。

为了帮助你理解，我们需要先来看看数据请求和传输流程图。就用常见的 HTTP 协议的 Request 和 Response 方式来描述，如下图所示。



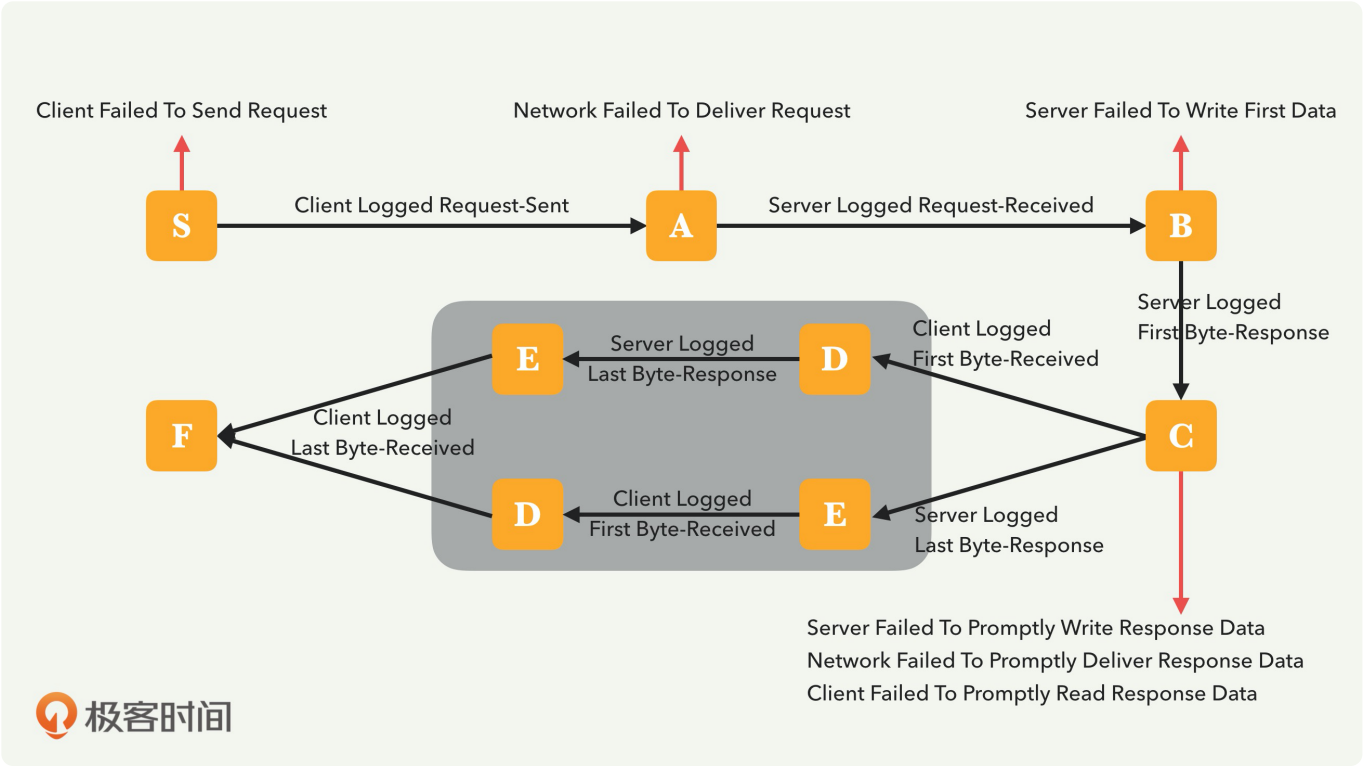
当客户端需要下载服务器的数据时，首先在 T0 发出数据请求；网络将请求发送到服务器后，服务器在 T1 收到数据。

然后，服务器开始准备数据，数据准备好后，服务器将开始在 T2 时发回数据。通过一系列 write() 调用。发送在 T4 完成。网络传输后，客户端在 T3 开始接收数据，并在 T5 完成接收。请注意，尽管其他时间戳是按照严格的顺序，T3 和 T4 的顺序可能会因实际情况而异。具体来说，对于小数据传输时，T4 可以先于 T3，因为单个 write() 调用就足够了。对于大数据传输，通常使用 T3 在 T4 之前。

接下来，我们来看看基于状态机的解决方案，它是一个针对 HTTP 数据传输问题的，完整而具体的解决方案。

从上面的过程中，我们可以看到，如果服务器无法接收到数据请求，则数据传递将不会发生，因此不会完成。

我用下图来表示整个状态机。这个状态机展示了整个 HTTP 数据传输的过程，包括 Request 和 Response。如果数据传递成功，状态机最后会到达状态 F。



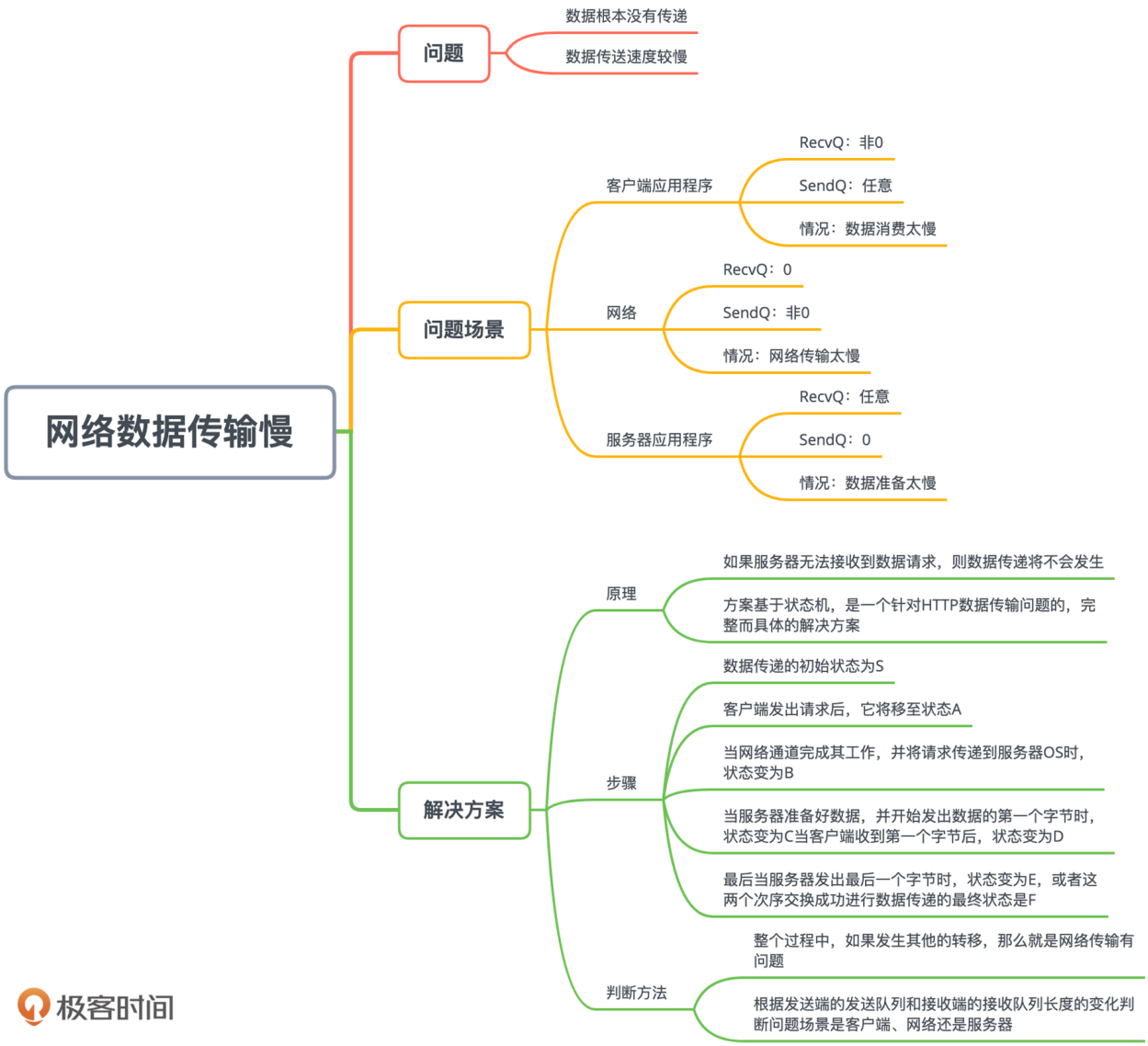
如上图，数据传递的初始状态为 State-S。客户端发出请求后，它将移至状态 A；当网络通道完成其工作，并将请求传递到服务器 OS 时，状态变为 B。当服务器准备好数据，并开始发出数据的第一个字节时，状态变为 C。

当客户端收到第一个字节后，状态变为 D；最后当服务器发出最后一个字节时，状态变为 E。或者这两个次序交换，成功进行数据传递的最终状态是 State-F。

在整个过程中，如果发生其他的转移，那么就是网络传输有问题了。我们就可以根据发送端的发送队列和接收端的接收队列长度的变化，轻松判断是谁的问题，比如是客户端，服务器或是网路。

总结

今天我们讲述了，互联网服务在传输数据时，如果发生传输速度太慢的问题，怎样才能快速地诊断到底是客户端、服务器端，还是网络的问题。



唐代诗人高适的《燕歌行》有几句诗：“山川萧条极边土，胡骑凭陵杂风雨。战士军前半死生，美人帐下犹歌舞。”说的是前方的战士，在前线出生入死；后方却有人逍遥自在的观赏美人歌舞，醉生梦死。这种冰火两重天的讽刺，部分原因，就是责任没有分清，以至于滥竽充数者可以逍遥自在。

对待数据传输缓慢问题，我们也很希望能快速地搞清责任，分清是哪里的问题，然后才能有针对性地继续分析。我们的解决方案就是根据 TCP 的 Send 和 Receive 队列大小变化，来快速诊断的方案。它能智能而快速地分清问题的大致范围：就是数据发送方、数据接收方，还是网络。

思考题

根据你平时的观察，公司业务有没有发生数据传输太慢的问题？如果发生了，你们一般怎么根因呢？如果采用本讲的思路，会不会更加快速地诊断？

欢迎你在留言区分享自己的思考，与我和其他同学一起讨论，也欢迎你把文章分享给自己的朋友。

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇　27 | 多任务环境中的Java性能问题，怎样才能不让程序互相干扰？

下一篇　29 | 如何彻底发挥SSD的潜力？

精选留言

写留言

> 由作者筛选后的优质留言将会公开显示，欢迎踊跃留言。